Journal of Soft Computing in Civil Engineering 8-3 (2024) 87-106

Contents lists available at SCCE

Journal of Soft Computing in Civil Engineering

Journal homepage: www.jsoftcivil.com

Generation of Dimensioned Floor Plans for a Given Boundary Layout

Pinki Pinki [1,*]; Krishnendra Shekhawat [2]; Dipam Goswami [3]; Ujjwal Jain [3]

1. Post Doctoral Fellow, Birla Institute of Technology and Science, Pilani, India
2. Associate Professor, Faculty of Mathematics, Birla Institute of Technology and Science, Pilani, India
3. Undergraduate Student, Birla Institute of Technology and Science, Pilani, India

Corresponding author: *456pinkiyadav@gmail.com*

https://doi.org/10.22115/SCCE.2023.377613.1583

ARTICLE INFO

Article history:
Received: 27 December 2022
Revised: 08 August 2023
Accepted: 31 December 2023

Keywords:
Rooms;
Non-rectangular floor plan;
Slicing tree;
Algorithm;
Dissection.

ABSTRACT

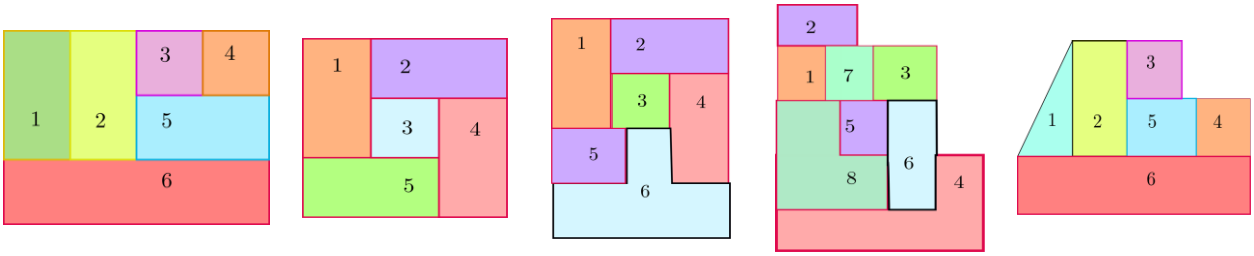
In literature, the generation of floor plans has mainly been confined to dimensionless floor plans with rectangular boundaries having no open spaces within the floor plan. In this paper, the user is allowed to construct a dimensioned boundary using slant, horizontal and vertical line segments, where dimensioned open spaces can be drawn within the boundary layout. Once the boundary is finalized, it can be partitioned into dimensioned blocks using vertical and horizontal dissections. Each block can be further partitioned into dimensioned rooms which results in a dimensioned floor plan F for the given boundary layout. The dissection method employed is based on the slicing tree approach, which results in floor plans that are amenable to slicing and consist of non-rectangular rooms, offering the potential for open spaces within floor plan F. As a preliminary step towards automation, we have developed an interactive user interface for generating dissected dimensioned floor plans.

How to cite this article: Pinki, P., Shekhawat, K., Goswami, D., Jain, U. Generation of dimensioned floor plans for a given boundary layout. J Soft Comput Civ Eng 2024;8(3):87–106. https://doi.org/10.22115/scce.2023.377613.1583

2588-2872/ © 2024 The Authors. Published by Pouyan Press.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

A *floor plan* (FP) is defined as a closed polygon divided by straight-line segments into a finite number of non-overlapping polygons called *rooms*. The shapes of rooms and floor plan boundary can be used to classify FPs. A FP is called a *rectangular floor plan* (RFP) if it has a rectangular boundary and consists of rectangular rooms. A RFP has been classified as slicing FPs and non-slicing FPs. A slicing FP is a rectangular dissection obtained by dissecting a rectangle horizontally or vertically into two smaller rectangles in a recursive manner (Figure 1a). An ordered tree, also known as a slicing tree [1], is the earliest representation used by researchers to represent slicing FPs in the literature [2]. Each internal node represents a vertical or a horizontal merge operation on the two descendants, while each leaf of the slicing tree corresponds to a block. A FP is called an *orthogonal floor plan* (OFP) if its boundary is rectangular, and it has rectangular and rectilinear rooms (at least one rectilinear). A *non-rectangular floor plan* (NRFP) has a rectilinear or slanted boundary. A *rectilinear floor plan* is a specific case of NRFP having a rectilinear boundary and rectangular or rectilinear rooms (Figure 1).



(a) Slicing floor plan (b) Non-slicing floor plan (c) Orthogonal floor plan (d) Rectilinear floor plan (e) Non-rectangular floor plan

Fig. 1. Types of Floor Plans.

In recent years, researchers have focused on the automated generation of floor plans. The generation of a floor plan is an important step in the design process and it has different problem perspectives. From an architectural perspective, it is a design problem treated as a preliminary layout which can be further modified and adjusted by architects. From a mathematical perspective, it is a polygonal representation of vertices which satisfies the adjacency requirements of a given graph. From a VLSI perspective, it is an optimization problem where it is required to optimize the chip area while structuring its design. Clearly, the problem of generating floor plans is of an interdisciplinary nature, which reflects the growing interest of researchers in the automated generation of floor plans. Therefore, there exist various approaches to automate the generation of floor plans which are based on graph theory, optimization, dissection, shape grammar, neural networks, machine learning and artificial intelligence, etc. At the same time, none of the approaches can be used to generate floor plans for a given boundary layout, in particular non-rectangular.

In this paper, we provide an interactive tool for the generation of dimensioned floor plans by dissecting a given plot area. Initially, a user-specific non-rectangular boundary layout has been created from a given rectangular plot. The user has the autonomy to determine the dimensions of the boundary which is made up of slant, horizontal and vertical line segments as shown in Figure

2a. Within the boundary, the user has a choice to insert dimensioned open spaces by selecting the interior points. After finalising the boundary, the layout can be partitioned into dimensioned blocks using vertical and horizontal dissections as shown in Figures 2b and 2d.

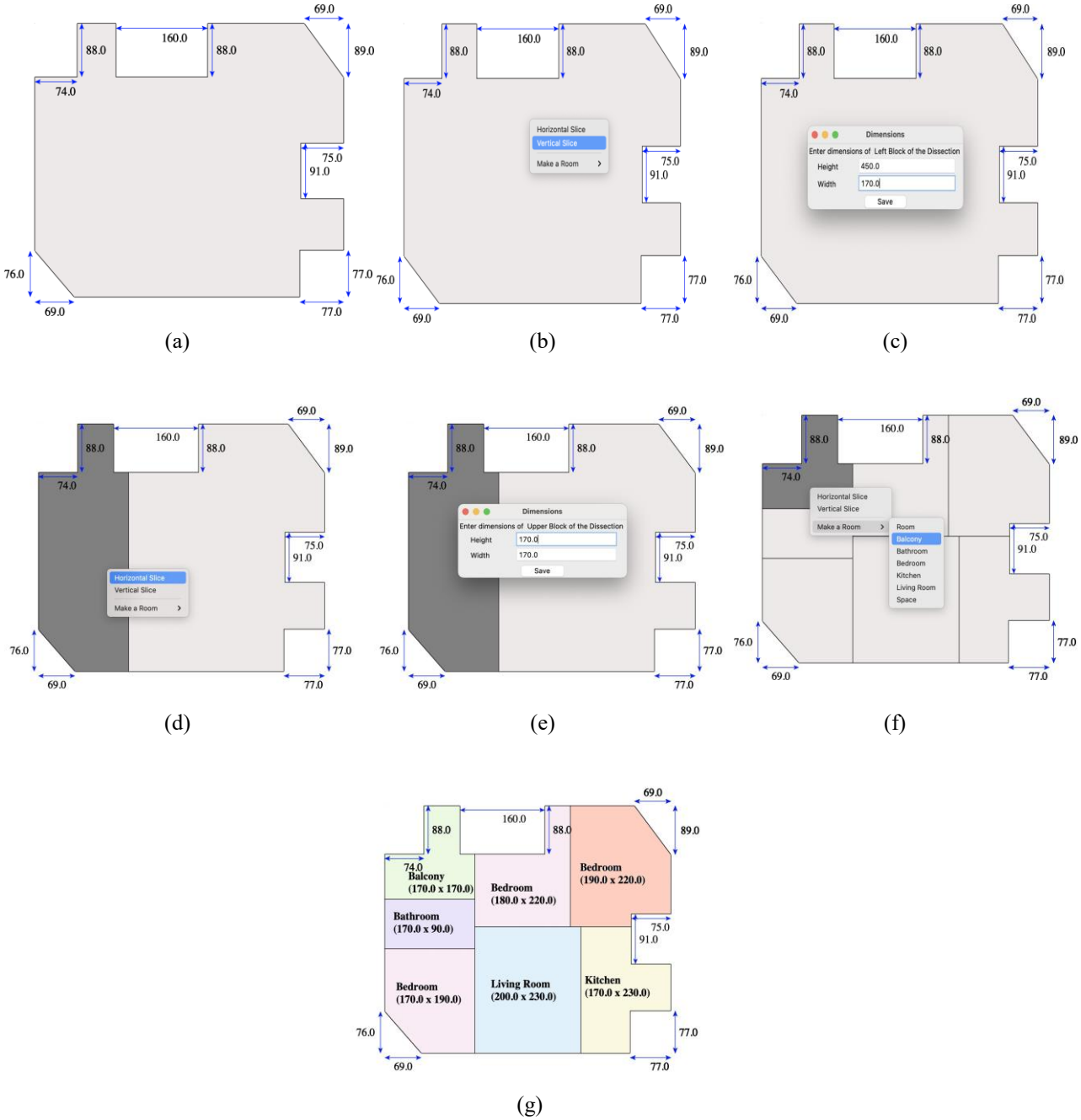


Fig. 2. Customised generation of a dimensioned floor plan using dissection method (a) A given layout with dimensioned non-rectangular boundary, (b) and (d) Types of dissection (horizontal or vertical dissection) required to create rectilinear/rectangular rooms, (c) and (e) Input dimension based on the dissection type for creating rectilinear/rectangular rooms, (f) Selecting a block to make it a room, (g) Required floor plan.

After slicing a block horizontally, let the newly formed blocks be A and B, where block A is the upper block and block B is the lower block. For horizontal slicing, the user has to input the height of the upper block A, since its width must be equal to the width of the parent block to ensure the generation of a FP as shown in Figures 2d and 2e. Similarly, after slicing a block vertically, let the newly formed blocks be C and D, where block C is the left block and block D is the right block. For vertical slicing, the user has to input the width of the left block C as shown in Figures 2b and 2c. For each block, the user has a choice to either consider it as a room or further partition it into dimensioned rooms as per requirements, which results in a dimensioned floor plan for the given boundary layout (refer to Figure 2 for the step-by-step explanation of the construction of dimensioned FP from a non-rectangular boundary). The proposed algorithm is implemented using Python programming language and it can be used for customized generation of dimensioned RFPs, OFPs, and NRFPs, with or without open spaces. Our work is the first step towards automating the generation of dimensioned floor plans for given non-rectangular boundary layouts, where adjacency relations would be incorporated in future work.

The rest of the paper is structured as follows. Section 2 examines the related work. Section 3 explains the concepts and definitions used in our approach. Section 4 presents the construction of dimensioned FPs with or without open spaces and provides an explanatory example. This section also provides the pseudocode for the proposed algorithm. Section 5 describes the results and discussion part along with the regeneration of some existing FPs. Section 6 concludes the paper and proposes future scope.

2. History of floor plans

The term “floor plan generation” has piqued the interest of researchers due to its broad range of applications in different fields such as architecture, VLSI, etc. Due to the growing attention, the problem has been intensively studied and different methodologies for the generation of FPs have been used in literature.

2.1. Generation of floor plans

Graph-theoretic approaches: The problem of generating FPs using graph theory has been initiated in 1960s by Levin [3]. He proposed a representation for an adjacency graph in the form of a FP. In 1970s, the problem has been extensively studied [4–8]. The problem of representing a graph in the form of a FP is mathematically challenging because not all graphs can have a FP (for example, non-planar graphs). Hence, it has been addressed on the basis of classes of graphs (i.e., outerplanar graphs[1], maximal planar graphs (MPGs)[2], planar triangulated graphs (PTGs)[3], 1-connected PTGs[4], bi-connected PTGs[5] etc.) and types of FPs (RFP, OFP, rectilinear FP). In 1985,

[1] An outerplanar graph is a graph that can be embedded in the plane with all of its vertices lying on the exterior face.

[2] A maximal planar graph (MPG) is a planar graph whose every face is a triangle and adding an edge to it would destroy its planarity.

[3] A planar triangulated graph (PTG) is a planar graph with no simple cycles of length 4 or more.

[4] A PTG with cut vertices.

[5] A PTG with no cut vertices.

Koźmiński and Kinnen gave a necessary and sufficient condition for the existence of a RFP for a given PTPG[6]. There exist many other approaches to generate RFPs, i.e., coloring technique [9], regular edge labelling [10], slicing tree [11,12], constraint satisfaction [13] and Monte-Carlo Tree search-based reinforcement learning [14], etc. Other works on the generation of RFPs can be found at [10,15–23].

In 1993, Yeap and Sarrafzadeh [24] discovered that there exists a graph for which RFPs do not exist. In the later years, a lot of research has been conducted on OFPs (rectilinear representations) for MPGs using different approaches such as face area concept, canonical labelling, duality, etc. [25–34]. Recently, the work presented by Shekhawat et al. [35] demonstrated the utilization of graph algorithms for the generation of OFPs with room shapes tailored according to user specifications.

Due to the applications of FPs in VLSI circuits and architectural layout arrangements, FPs are not limited to rectangular boundaries only. In 1982, Baybars et al. [36] proposed the generation of FPs with rectilinear boundaries and circulation spaces for a given MPG using the concept of pseudo-geometric dual. In 1992, Nummenmaa [37] gave an algorithm to construct rectilinear FPs for MPGs using the canonical representation of planar graphs. In 2020, Wang et al. [38] proposed FP generation with rectilinear boundaries and rooms. In 2022, Sumit et al. [39] presented the automated generation of dimensioned and dimensionless FPs. They proposed a software GPLAN which uses graphic theoretic algorithms along with linear optimization techniques. In a recent study, Raveena et al. [40] introduced the necessary and sufficient conditions for the existence of non-trivial L-shaped FPs corresponding to PTPGs and gave an algorithm for its construction, if it exists.

Other approaches: In the last few decades, researchers have explored different approaches for the automatic generation of FPs. In 2010, Merrell et al. [41] introduced supervised learning for architectural design and proposed the automated generation of FPs using Bayesian networks. In 2013, [42] devised an enhanced hybrid evolutionary computation scheme for generating FPs that combines an Evolutionary Strategy (ES) and a Stochastic Hill Climbing (SHC) technique. In 2018, Wu et al. [43] presented the generation of dimensioned FPs using mixed quadratic programming. There are many other approaches for the generation of FPs, i.e., neural networks [44], generic optimization [45,46], artificial intelligence [47], data-driven techniques [48,49], constrained optimization [50], generative adversarial layout refinement network [51], integer linear programming [52], Shape grammar [53–55] etc. For more work on the generation of rectilinear FPs corresponding to the user-specified design requirements, refer to [38,44,48,56]. In 2023, Wang et al. [57] proposed a framework for automated building layout generation, leveraging the power of deep learning and graph algorithms. On the other hand, Weber et al. [58] conducted a recent study that critically reviewed various methods and their applications for automating architectural FP design.

[6] A properly triangulated planar graph (PTPG) is a connected planar graph where all interior faces have length 3, internal vertices have degree ≥ 4 and non-face cycle with length ≥ 4.

Dimensioned FPs: Ideas for the work on dimensioned FPs were proposed in 1970s by Cousin [4]. In 1982, Roth et al. [9] proposed dimensioned FPs for an adjacency graph using the PERT algorithm. In 1987, Rinsma [16] presented the dimensioned RFPs and OFPs for a given tree. Many researchers used weighted graphs as an input to generate dimensioned FPs [19,30,59]. For recent work on dimensioned FPs, refer to [38,60,61].

2.2 Significance of floor plans with non-rectangular boundary layout and proposed work

In modern times, design cannot remain restricted to rectangular or rectilinear boundaries. It can be observed that for complex building structures like schools, hospitals and office buildings etc., we may require layouts with non-rectangular boundary having open spaces within the layout. Hence, this work is a primary step towards the automated generation of floor plans with non-rectangular boundaries.

By reviewing the existing work on FPs, it can be observed that the construction of FPs is mainly restricted to FPs with rectangular or rectilinear boundaries. Also, the work on the rectilinear boundary is restricted to specific shapes like L-shape, T-shape, staircase, etc. (refer to [33,38,43,44,48,56]) with no internal spaces (atrium) within the FP. Also, there is no work on FPs which consider slanted line segments for a boundary layout. In this work, we propose a graphical user interface for generating the initial architectural dimensioned layout of a FP with no restriction on the boundary, i.e., it is feasible to consider a dimensioned plot with a rectangular or non-rectangular boundary or slanted boundary. Also, it is possible to add dimensioned atrium within the plot area. Using the dissection method, the required plot would be partitioned into the required number of dimensioned rooms to have a required floor plan.

3. Preliminaries

Here, we cover basic definitions and their notations which are used frequently in this paper. These terms have been previously mentioned in literature [62] and are explained here for a better understanding.

Definition 1 [FP]: A floor plan (FP) is a closed polygon partitioned into a finite number of non-overlapping polygons by straight/slant line segments. A room in a FP is a polygon that cannot be further subdivided. A FP with rectangular boundary and rectangular rooms is called a rectangular floor plan (RFP). RFPs are mainly categorised into the following two types:

i. Slicing FP

ii. Non-slicing FP

Definition 2 [SFP]: Slicing floor plan (SFP) is obtained by finitely repeated bisection of a larger rectangle until each part is a single rectangle (Figure 1a). It can be represented by a rooted binary tree known as a slicing tree (ST). Leaves of a ST represent rooms of a FP and its internal vertices represent cut types indicated by H (for horizontal division) and V (for vertical division). For example, in Figure 3, a SFP and its corresponding ST are given. There can be more than one ST corresponding to a FP (Figure 4).

Definition 3 [NSFP]: A Non-slicing floor plan (NSFP) is a more general FP. It cannot be bisected and hence cannot be represented by ST.

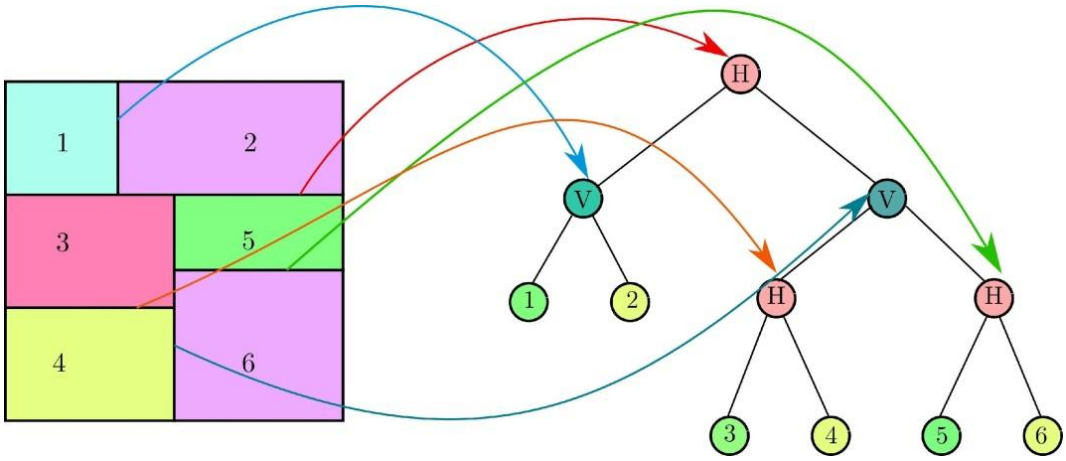


Fig. 3. A slicing floor plan and its corresponding slicing tree.

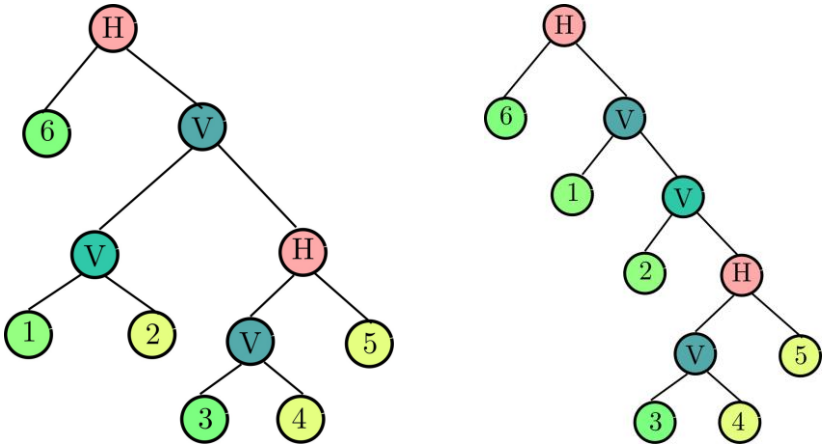


Fig. 4. Different representations of a Slicing Floor Plan given in Figure 1(a).

Definition 4 [Block]: A block in a FP is a polygon which has either a vertical or a horizontal division that divides the block into two smaller blocks, i.e., child blocks. If a block has a vertical division, then its child block has the same height as that of the parent block. Similarly, if a block has a horizontal division, then its child block has the same width as that of the parent block. A FP is a block in itself.

Notations:

FP: floor plan, RFP: rectangular FP, OFP: orthogonal FP, NRFP: non-rectangular FP, SFP: slicing FP, NSFP: non-slicing FP, MPG: maximal planar graph, PTG: planar triangulated graph, PTPG: properly triangulated planar graph.

4. Construction of dimensioned FPs with or without open spaces

This section describes the working of the proposed algorithm which is implemented using Python programming language. The proposed algorithm is used for the customized generation of

a dimensioned FP starting from a given non-rectangular dimensioned layout and gradually dissecting it horizontally and vertically. Its steps are as follows:

Step 1: Creating a rectangular plot

To create a rectangular plot, the user needs to enter its width and height, which generates a rectangular plan on the screen with the required width and height (Figure 5).

Fig. 5. Entering dimensions of the required rectangular plan and corresponding obtained plan.

Step 2: Forming a dimensioned boundary layout

Step 2.1: Selection of boundary points

To form a FP boundary, the user needs to select points on the boundary of the rectangular plot with two choices resulting in slanted line segments or rectilinear line segments (Figure 6).

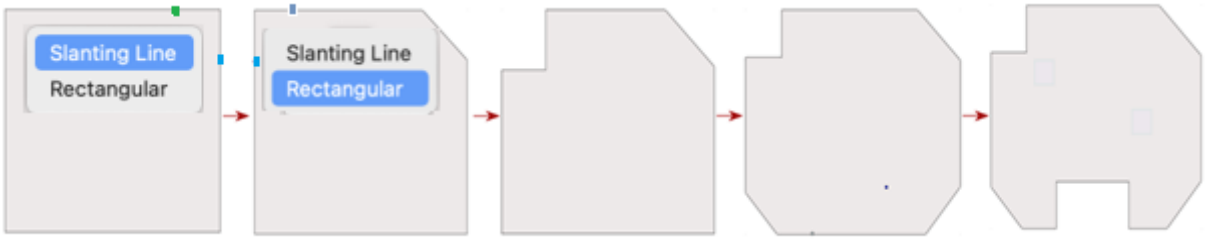
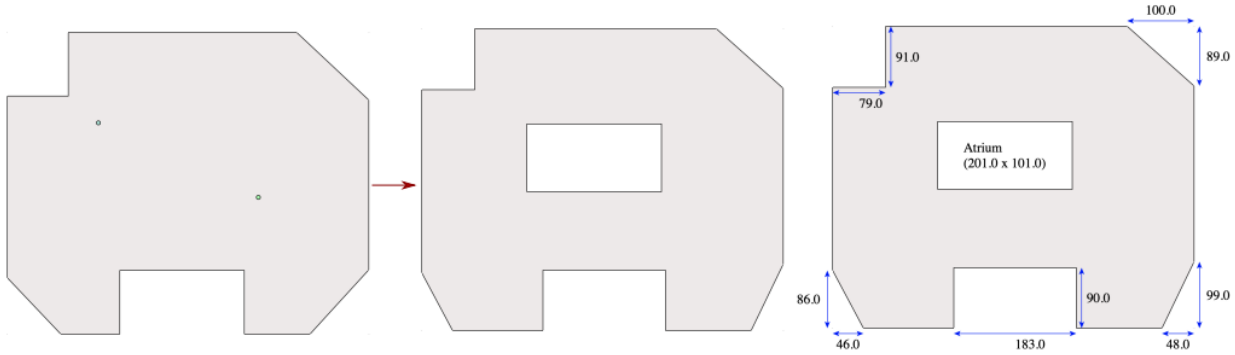


Fig. 6. Selection of boundary points.



(a) (b)

Fig. 7. (a) Selection of interior points, (b) Finalised non-rectangular dimensioned boundary layout with atrium.

Step 2.2: Selection of interior points

Selecting the points inside the plan creates open spaces (Figure 7). After finalizing the boundary points and interior points, a dimensioned boundary layout is obtained as shown in Figure 7(b).

Step 3: Constructing a FP using dissection

To construct a required FP, the user can sequentially dissect the plan obtained in Step 2, either horizontally or vertically, by right-clicking on a single block at a time. After each dissection, a block area gets partitioned into two sub-blocks. If the user does not want to further partition a block, then they have the option to designate the block as a room.

Step 3.1: Vertical and horizontal dissection

Each block in a FP is considered as a node in the slicing tree representation of the FP. Every block has the following attributes: parent, left_child, right_child, height, width, slice_type, d1, d2, d3 and d4. The attributes parent, left_child, and right_child point to another block in the FP, height and width store dimensions of a particular block, slice_type stores the type of slicing on its parent block and d1, d2, d3, d4 stores the top-left corner x-coordinate, top-left corner y-coordinate, bottom right corner x-coordinate, bottom right corner y-coordinate of the block respectively. The attributes for all the blocks are stored independently for each of them. On dissecting a block, it gets connected to its left child and right child representing the two blocks formed as a result of dissection.

Step 3.2: Block dimensioning

For horizontal slicing, the user has to input the height of the upper block of dissection since its width equals the width of the parent block to ensure the generation of a FP. We obtain the height of the lower block by subtracting the height of the upper block from the height of the parent block. Similarly, for vertical slicing, the user has to input the width of the left block of dissection. Two new blocks formed by dissection have the same parent. The left_child and right_child properties of the parent block now point to the new blocks. The height and width of the new blocks are set to the input dimensions. The d1, d2, d3, and d4 values of these blocks are set according to their positions in the FP. These values increase on moving downward and rightward (refer to Figure 8 for a better illustration of the positioning of blocks in a FP).

Step 3.3: Formation of rooms

If the user chooses a block to make it a room, then this block is considered as a leaf of the slicing tree (refer to Figure 9(a) for an illustration of the step where the choice is given for making the block as a room). A room is represented by a shaded rectangle and labeled with a room number where the dimensions of the room are displayed below the room name (refer to Figure 9(b) which is a dimensioned FP and for the illustration of all the steps for the transformation of a plot into a dimensioned FP, refer Figures 5 to 9).

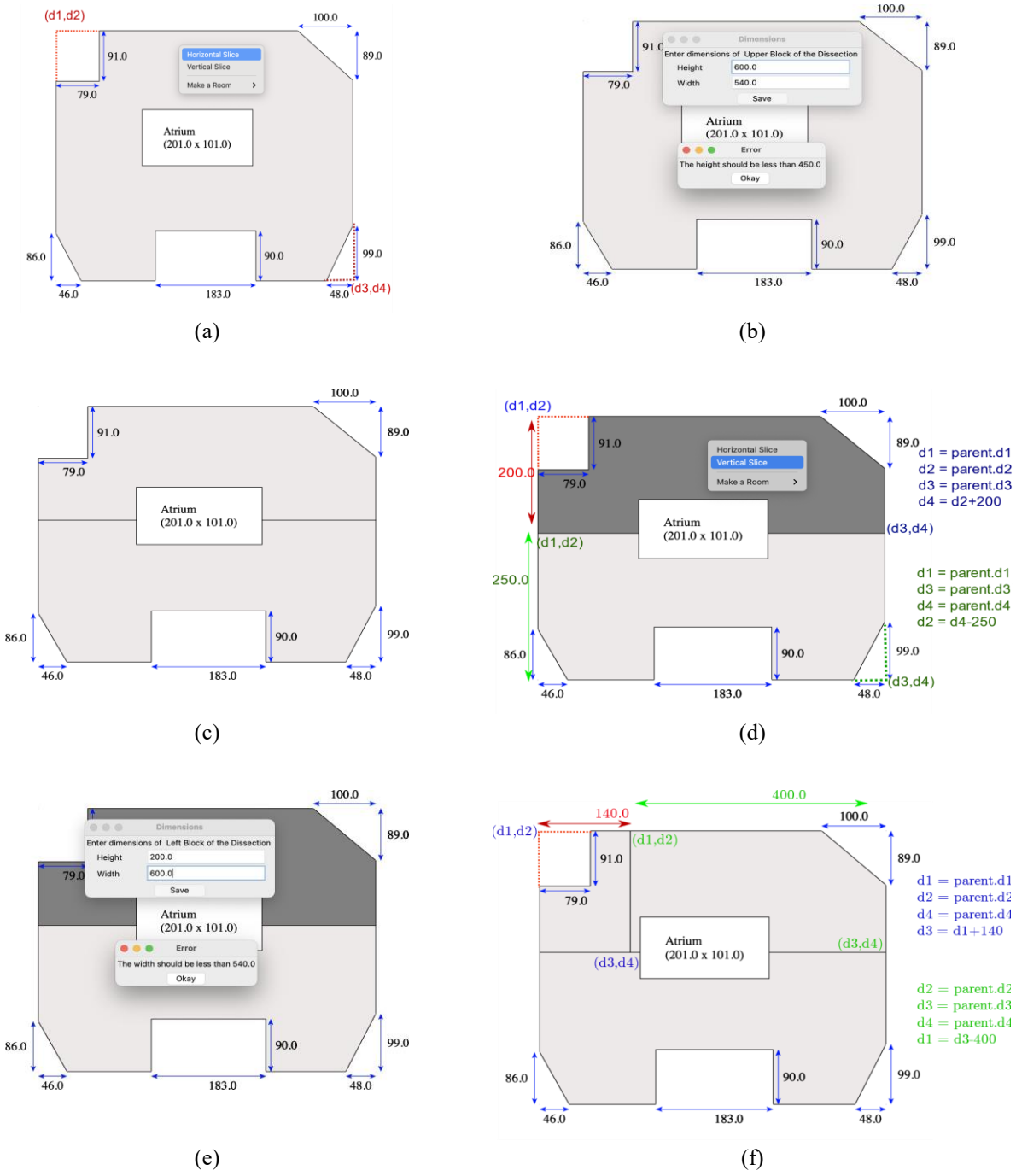


Fig. 8. (a) Choice for the type of dissection (b) After opting for horizontal dissection, the dimension of the block needs to be given as input, which would show an error if horizontal dissection is not possible for the provided dimensions, (c) Horizontal dissection of the block satisfying dimensions as given in (b), (d) Again providing the choices for the type of dissection,(e) After opting for vertical dissection, dimension of the block needs to be given which would show an error if the vertical dissection is not possible for provided dimensions, (f) Vertical dissection of the block satisfying dimensions given in (e). (a) Selection of interior points, (b) Finalised non-rectangular dimensioned boundary layout with an atrium.

4.1. Pseudo code for the construction of dimensioned FPs

In this section, we present pseudo code for the generation of FPs which is implemented using Python programming language with the help of its GUI library Tkinter.

Algorithm 1. FP

Input: A rectangular plot and a non-rectangular boundary layout *A*

Output: A dimensioned floor plan

Data: Dimension of a floor plan and its dissected components

1 Create a rectangular plot with given dimensions (Width of plot > 0, Height of plot > 0)

2 Finalise the type of FP

(a) select external points for having boundary with rectilinear and slant line segments

(b) select points inside the plot for creating atriums

3 while *(all blocks are not rooms)* **do**

4 Choose the block of a plot to dissect

5 Choose vertical or horizontal dissection for the chosen block of a plot

6 **if** *Slice type is vertical* **then**

7 Height for left sub-block = Height of block

8 Width for left sub-block = Input by user

9 **if** *Width of left sub-block* < *Width of block* **then**

10 Partition the selected block vertically with user input dimensions

11 **else**

12 Dissection is not feasible with given dimensions

13 Width has to be less than width of the block

14 Enter width for left sub-block

15 **else if** *Slice type is horizontal* **then**

16 Width for upper sub-block = Width of block

17 Height for upper sub-block = Input by user

18 **if** *Height of upper sub-block* < *Height of block* **then**

19 Partition the selected block horizontally with user input dimensions

20 **else**

21 Dissection is not feasible with given dimensions

22 Height has to be less than height of the block

23 Enter height for upper sub-block

24 **else**

25 Assign room name

26 Obtained plan is the desired dimensioned FP

27 Exit

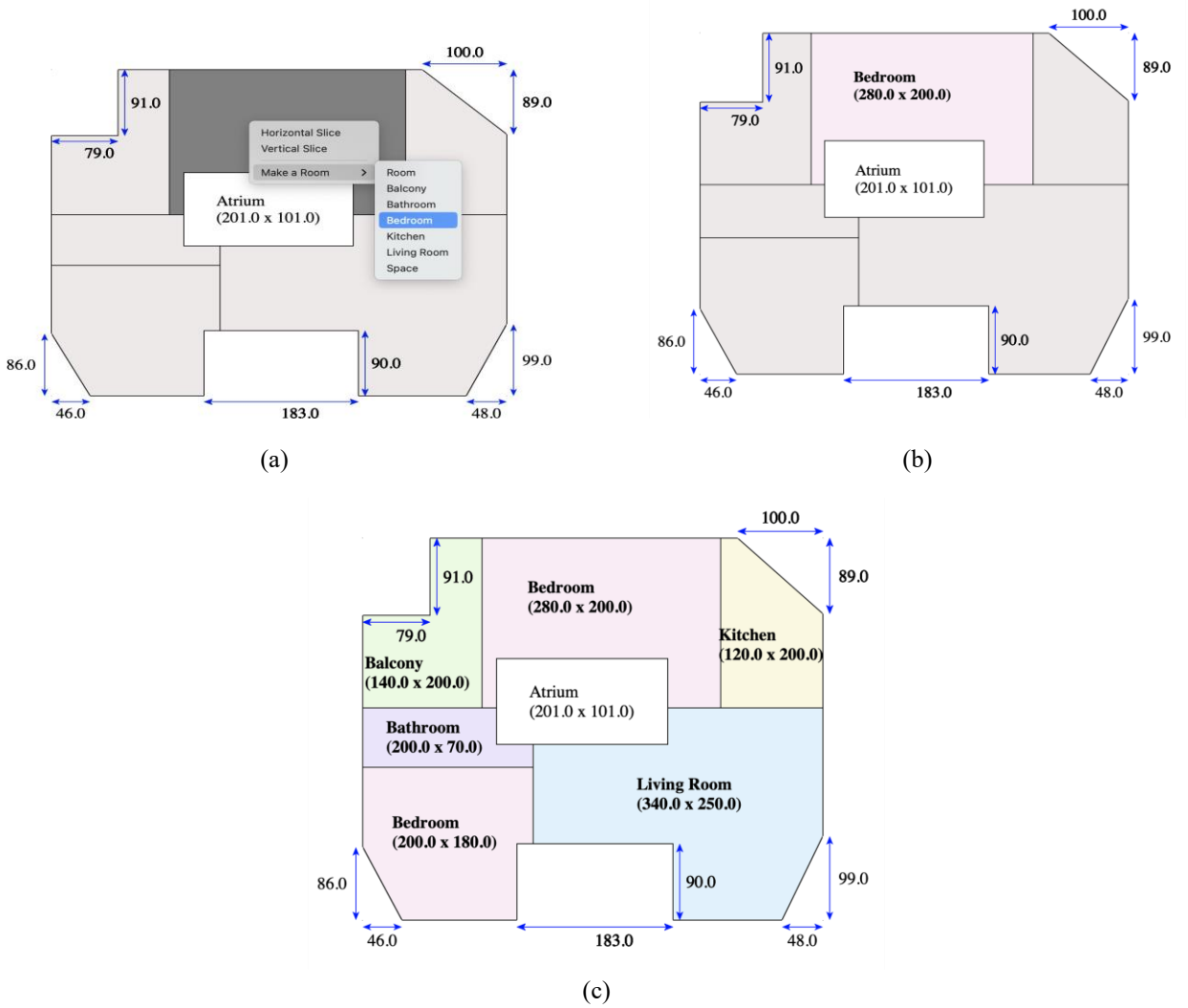


Fig. 9. (a) Choice for the type of rooms, (b) Formation of rooms from the dissected blocks, (c) Obtained dimensioned FP.

5. Results and discussion

In this section, we show some instructive examples that give a quick overview of the proposed prototype by demonstrating various types of dimensioned FPs that it can generate. This section also shows how it can be used to regenerate various existing floor plans.

5.1. Regeneration of existing RFP using our prototype:

1. Regeneration of Plan Topology of Villa Badoer

The generated FP maintains the following constraints given in [63] (Figure 10):

(i) The sala ‘R5’ is the largest room.

(ii) The aspect ratio of any room is less than or equal to 2:1 except ‘R7’.

(iii) Rooms surrounding the sala are not identical in shape.

(iv) The ratio of the area of the largest room, the sala to that of the smallest room is 9:1.

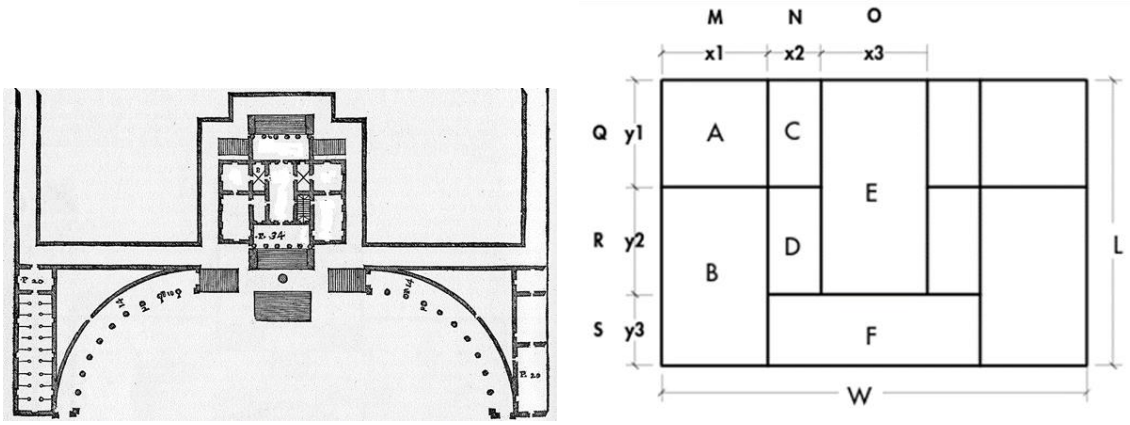
(v) No room is smaller than 40 square feet.

(vi) Width of the cells in the underlying grid lying on the axis of symmetry is wider than adjacent cells.

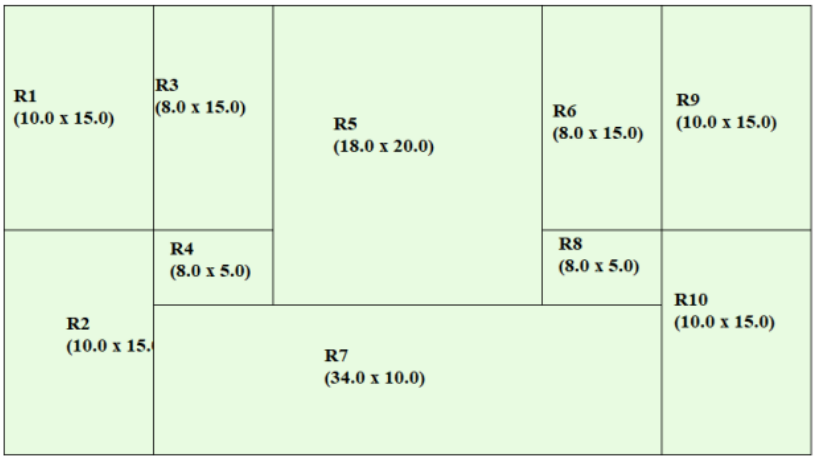
(vii) No room has dimensions less than 7 feet.

2. **Regeneration of Plan Topology of Palladio's Palazzo Della Torre [58]**

Palladio's Palazzo Della Torre is the fourth building in Andrea Palladio's Four Books on Architecture, Book II. Count Giovanni Battista Della Torre of Verona, Italy, commissioned it in 1555 but died before Palazzo could be completed. Here, dimensions are taken based on the representation of the aspect ratio in the existing plan.



(a)



(b)

Fig. 10. Floor Plan Topology of Villa Badoer[7] (a) Existing Floor Plan and its topology [63] (b) Regenerated Floor Plan.

[7] http://www.projekte.kunstgeschichte.uni-muenchen.de/arch_complete_vers/40-ren-barock-architektur/studieneinheiten/lektion_11/XI_2_10.htm

5.2. Explanatory examples

Using the prototype presented in this paper, different types of FPs can be generated (Figure 12). Some of them are given as follows:

1. **Type 1:** NRFP with rectilinear boundary and rectilinear rooms with no open spaces (Figure 12b)

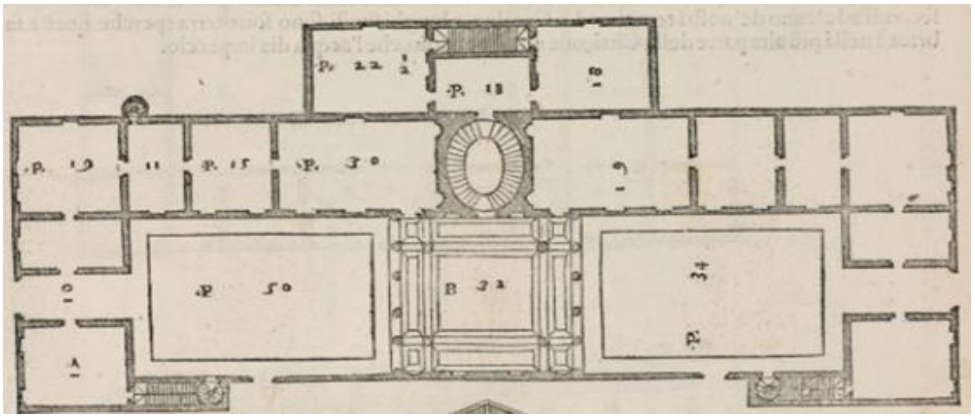
2. **Type 2:** NRFP with rectilinear boundary, rectilinear rooms and with open spaces (Figure 12c)

3. **Type 3:** NRFP with slanted boundary and no open spaces (Figure 12a)

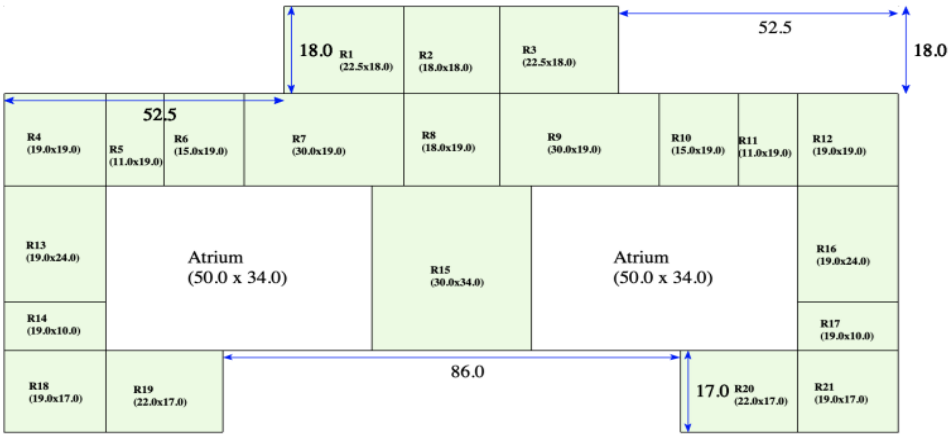
4. **Type 4:** NRFP with slanted boundary and open spaces

5. **Type 5:** NRFP with the combination of slanted and rectilinear boundary (Figure 12a, e, f)

6. **Type 6:** RFP (Figure 10)



(a)



(b)

Fig. 11. Floor Plan Topology of Palazzo Della Torre (a) Existing Floor Plan and its topology[8] (b) Regenerated Floor Plan.

[8] https://www.ribapix.com/Palazzo-Dalla-Torre-Verona-plan-and-facade_RIBA126891#

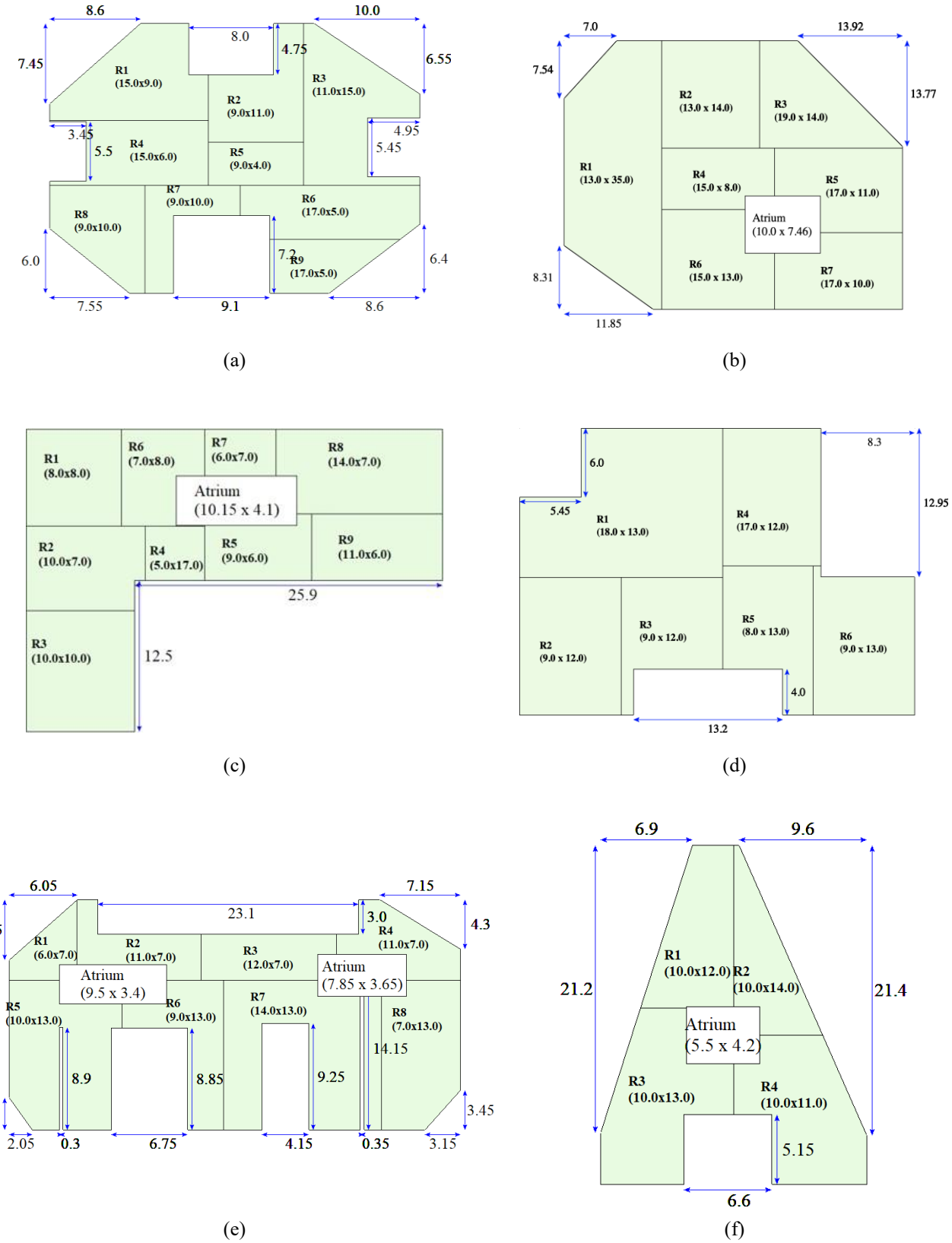


Fig. 12. Different types of FPs generated by the prototype.

6. Conclusion and future work

In this paper, we proposed a semi-automatic tool for the generation of dimensioned floor plans for a given boundary layout. The dimensions of the boundary are chosen by the user which is made up of slant as well as straight line segments. Within the boundary, there is a choice to insert dimensioned open spaces. At this stage, the tool could be helpful to architects where they have a choice to draw the layout boundary and then the boundary can be dissected by first partitioning it into different dimensioned blocks and further each block is dissected into required rooms. Here, a dissection is done by simply inputting the width and height of the required block. For each dissection, there is a choice of vertical as well as horizontal dissection.

The proposed work is the first work considering any dimensioned boundary layout for floor plan generation. Also, it presents a GUI developed in Python for making the work accessible to the users. At the same time, to make the tool more useful, we need to automate the process of dissection by considering certain architectural rules. For example, a bedroom must have a window and it must be at least 8 feet wide, i.e., we cannot allow the user to make the bedroom as an interior room. The next challenge will be to consider the adjacency requirements for the required rooms. Limitations will be discussed in detail in the next section. A brief outline of the proposed work is as follows:

1. Within existing literature, the floor plans having boundaries made up of slant lines and the inclusion of atriums have not been explored. This study marks an initial stride toward the automated generation of dimensioned floor plans within any given arbitrary non-rectangular boundary. Clearly, the proposed methodology offers a diverse spectrum of choices to users for space allocation within a given layout, facilitating the creation of non-rectangular dimensioned floor plans with desired shapes, as depicted in Figure 12.

2. The prototype introduced in this paper employs the dissection method, ensuring that the generated blocks or rooms adhere to the user's designated dimensional constraints. It also possesses the ability to generate symmetric dimensioned floor plans (as shown in Figures 10 and 11). Furthermore, it offers a range of alternative layouts, as each block possesses the flexibility to incorporate spaces in multiple directions, allowing for extensive customization to suit specific requirements.

The problem of generating dimensioned FPs corresponding to a given adjacency graph with arbitrary input non-rectangular boundary layout is mathematically very challenging, which we aim to address in future.

7. Limitations

The study represents an initial stride towards the automated generation of dimensioned floor plans with non-rectangular boundaries. It exhibits certain limitations that are earmarked for future refinement.

The prototype necessitates user involvement in determining where dissections should occur to ensure that the area requirements of individual rooms are satisfied. This entails providing dimensions for each slicing, thus requiring user input for customization. This constraint could potentially be mitigated by taking room dimensions as input and subsequently automating the dissection process.

Furthermore, there are several promising avenues for future research and development, including:

1. Exploring the generation of floor plans by considering aspect ratios as input rather than taking dimensions for every block.
2. Investigating the feasibility of generating floor plans by employing a dissection tree as input which introduces adjacency constraints to the problem.
3. Extending the capabilities of the prototype to generate floor plans with circulations. A circulation is an interior space of a FP through which people can move. In future, we try to generate FP with the shortest circulation path spanning all the rooms with one or more entrances.
4. Qualitative analysis of the obtained floor plans by architects.

Addressing these limitations and pursuing these avenues of exploration has the potential to significantly enhance the robustness and applicability of our approach.

Funding

This research received no external funding.

Conflicts of interest

The authors declare no conflict of interest.

Authors contribution statement

PP, KS, DG: Conceptualization; PP, DG: Roles/Writing – original draft; PP, DG, UJ: Software, Visualization; KS: Supervision; PP, KS: Writing – review & editing.

Supplementary material

Regarding the implementation of the proposed work in Python, a video showing the generation of dimensioned sliceable floor plan layout with rectilinear rooms and non-rectangular boundaries, with open spaces in the interior is available at https://www.dropbox.com/scl/fi/ilv1jph3jybw6s2aom0ny/RPF.mp4?rlkey=xumagnscv67xwgi521559m253&dl=0.

References

[1] Szepieniec AA, Otten RHJM. The genealogical approach to the layout problem. Proc. seventeenth Des. Autom. Conf. Des. Autom. - DAC ’80, New York, New York, USA: ACM Press; 1980, p. 535–42. https://doi.org/10.1145/800139.804582.

[2] Otten RHJM. Automatic Floorplan Design. 19th Des. Autom. Conf., IEEE; 1982, p. 261–7. https://doi.org/10.1109/DAC.1982.1585510.

[3] Levin PH. Use of graphs to decide the optimum layout of buildings. Archit J 1964;7:809–15.

[4] Cousin J. Organisation topologique de l'espace architectural= Topological organization of architectural spaces. (No Title) 1970.

[5] Grason J. A dual linear graph representation for space-filling location problems of the floor plan type. Emerg Methods Environ Des Plan 1970.

[6] Steadman P. Graph theoretic representation of architectural arrangement. Archit Res Teach 1973:161–72.

[7] Mitchell WJ. An approach to automated generation of minimum cost dwelling unit planes: International Technical Cooperation Centre, Tel-Aviv (ITCC) Review, Vol 4, No 3,(1975), pp 116–139. Comput Des 1977;9:69.

[8] Sauda EJ. Computer program for the generation of dwelling unit floor plans 1975.

[9] Roth J, Wachman A. A Model for Optimal Compact Packing of Convex Cells in an Ortogonal Network. Technion-IIT, Faculty of Architecture and Town Planning; 1978.

[10] Kant G, He X. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. Theor Comput Sci 1997;172:175–93. https://doi.org/10.1016/S0304-3975(95)00257-X.

[11] Del Río-Cidoncha G, Martínez-Palacios J, Iglesias JE. A multidisciplinary model for floorplan design. Int J Prod Res 2007;45:3457–76. https://doi.org/10.1080/00207540600889550.

[12] Marson F, Musse SR. Automatic Real-Time Generation of Floor Plans Based on Squarified Treemaps Algorithm. Int J Comput Games Technol 2010;2010:1–10. https://doi.org/10.1155/2010/624817.

[13] Regateiro F, Bento J, Dias J. Floor plan design using block algebra and constraint satisfaction. Adv Eng Informatics 2012;26:361–82. https://doi.org/10.1016/j.aei.2012.01.002.

[14] Shi F, Soman RK, Han J, Whyte JK. Addressing adjacency constraints in rectangular floor plans using Monte-Carlo Tree Search. Autom Constr 2020;115:103187. https://doi.org/10.1016/j.autcon.2020.103187.

[15] Bhasker J, Sahni S. A linear algorithm to find a rectangular dual of a planar triangulated graph. Algorithmica 1988;3:247–78. https://doi.org/10.1007/BF01762117.

[16] Rinsma I. Rectangular and Orthogonal Floorplans with Required Room Areas and Tree Adjacency. Environ Plan B Plan Des 1988;15:111–8. https://doi.org/10.1068/b150111.

[17] Lai Y-T, Leinwand SM. A theory of rectangular dual graphs. Algorithmica 1990;5:467–83. https://doi.org/10.1007/BF01840399.

[18] He X. On Finding the Rectangular Duals of Planar Triangular Graphs. SIAM J Comput 1993;22:1218–26. https://doi.org/10.1137/0222072.

[19] Eppstein D, Mumford E, Speckmann B, Verbeek K. Area-Universal and Constrained Rectangular Layouts. SIAM J Comput 2012;41:537–64. https://doi.org/10.1137/110834032.

[20] Shekhawat K, Duarte JP. Introduction to generic rectangular floor plans. Artif Intell Eng Des Anal Manuf 2018;32:331–50. https://doi.org/10.1017/S0890060417000671.

[21] SHEKHAWAT K, PINKI P. Construction of Architectural Floor Plans for Given Adjacency Requirements 2020.

[22] Kumar V, Shekhawat K. A transformation algorithm to construct a rectangular floorplan. Theor Comput Sci 2021;871:94–106. https://doi.org/10.1016/j.tcs.2021.04.014.

[23] Chaplick S, Kindermann P, Klawitter J, Rutter I, Wolff A. Morphing Rectangular Duals, 2023, p. 389–403. https://doi.org/10.1007/978-3-031-22203-0_28.

[24] Yeap K-H, Sarrafzadeh M. Floor-Planning by Graph Dualization: 2-Concave Rectilinear Modules. SIAM J Comput 1993;22:500–26. https://doi.org/10.1137/0222035.

[25] Sun Y, Sarrafzadeh M. Floorplanning by graph dualization: L-shaped modules. Algorithmica 1993;10:429–56. https://doi.org/10.1007/BF01891831.

[26] Liao C-C, Lu H-I, Yen H-C. Compact floor-planning via orderly spanning trees. J Algorithms 2003;48:441–51. https://doi.org/10.1016/S0196-6774(03)00057-9.

[27] de Berg M, Mumford E, Speckmann B. On rectilinear duals for vertex-weighted plane graphs. Discrete Math 2009;309:1794–812. https://doi.org/10.1016/j.disc.2007.12.087.

[28] Akbari Jokar MR, Shoja Sangchooli A. Constructing a block layout by face area. Int J Adv Manuf Technol 2011;54:801–9. https://doi.org/10.1007/s00170-010-2960-4.

[29] Ueckerdt T. Geometric representations of graphs with low polygonal complexity 2012.

[30] Alam MJ, Biedl T, Felsner S, Kaufmann M, Kobourov SG, Ueckerdt T. Computing cartograms with optimal complexity. Proc. twenty-eighth Annu. Symp. Comput. Geom., New York, NY, USA: ACM; 2012, p. 21–30. https://doi.org/10.1145/2261250.2261254.

[31] Chang Y-J, Yen H-C. Rectilinear Duals Using Monotone Staircase Polygons, 2014, p. 86–100. https://doi.org/10.1007/978-3-319-12691-3_8.

[32] Chang Y-J, Yen H-C. Constrained floorplans in 2D and 3D. Theor Comput Sci 2015;607:320–36. https://doi.org/10.1016/j.tcs.2015.07.063.

[33] Pinki P, Shekhawat K. Linear-time construction of floor plans for plane triangulations. Commun Comb Optim 2023;8:673–92. https://doi.org/10.22049/cco.2022.27814.1361.

[34] Pinki P, Shekhawat K. Characterization of Graphs Based on Number of Bends in Corresponding Floor plans. Proc. 6th Int. Conf. Algorithms, Comput. Syst., New York, NY, USA: ACM; 2022, p. 1–8. https://doi.org/10.1145/3564982.3564994.

[35] Shekhawat K, Lohani R, Dasannacharya C, Bisht S, Rastogi S. Automated generation of floorplans with non-rectangular rooms. Graph Models 2023;127:101175. https://doi.org/10.1016/j.gmod.2023.101175.

[36] Baybars I, Eastman CM. Enumerating architectural arrangements by generating their underlying graphs. Environ Plan B Plan Des 1980;7:289–310. https://doi.org/10.1068/b070289.

[37] Nummenmaa J. Constructing compact rectiliner planar layouts using canonical representation of planar graphs. Theor Comput Sci 1992;99:213–30. https://doi.org/10.1016/0304-3975(92)90349-K.

[38] Wang X-Y, Zhang K. Generating layout designs from high-level specifications. Autom Constr 2020;119:103288. https://doi.org/10.1016/j.autcon.2020.103288.

[39] Bisht S, Shekhawat K, Upasani N, Jain RN, Tiwaskar RJ, Hebbar C. Transforming an Adjacency Graph into Dimensioned Floorplan Layouts. Comput Graph Forum 2022;41:5–22. https://doi.org/10.1111/cgf.14451.

[40] Raveena, Shekhawat K. A theory of L-shaped floor-plans. Theor Comput Sci 2023;942:57–92. https://doi.org/10.1016/j.tcs.2022.11.025.

[41] Merrell P, Schkufza E, Koltun V. Computer-generated residential building layouts. ACM SIGGRAPH Asia 2010 Pap. - SIGGRAPH ASIA '10, New York, New York, USA: ACM Press; 2010, p. 1. https://doi.org/10.1145/1866158.1866203.

[42] Rodrigues E, Gaspar AR, Gomes Á. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests. Comput Des 2013;45:898–910. https://doi.org/10.1016/j.cad.2013.01.003.

[43] Wu W, Fan L, Liu L, Wonka P. MIQP-based Layout Design for Building Interiors. Comput Graph Forum 2018;37:511–21. https://doi.org/10.1111/cgf.13380.

[44] Hu R, Huang Z, Tang Y, Van Kaick O, Zhang H, Huang H. Graph2Plan. ACM Trans Graph 2020;39. https://doi.org/10.1145/3386569.3392391.

[45] Laignel G, Pozin N, Geffrier X, Delevaux L, Brun F, Dolla B. Floor plan generation through a mixed constraint programming-genetic optimization approach. Autom Constr 2021;123:103491. https://doi.org/10.1016/j.autcon.2020.103491.

[46] Fan Z, Liu J, Wang L, Cheng G, Liao M, Liu P, et al. Automated layout of modular high-rise residential buildings based on genetic algorithm. Autom Constr 2023;152:104943. https://doi.org/10.1016/j.autcon.2023.104943.

[47] He F, Huang Y, Wang H. iPLAN: Interactive and procedural layout planning. Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, p. 7793–802.

[48] Wu W, Fu X-M, Tang R, Wang Y, Qi Y-H, Liu L. Data-driven interior plan generation for residential buildings. ACM Trans Graph 2019;38:1–12. https://doi.org/10.1145/3355089.3356556.

[49] Lu Z, Wang T, Guo J, Meng W, Xiao J, Zhang W, et al. Data-driven floor plan understanding in rural residential buildings via deep recognition. Inf Sci (Ny) 2021;567:58–74. https://doi.org/10.1016/j.ins.2021.03.032.

[50] Para W, Guerrero P, Kelly T, Guibas L, Wonka P. Generative Layout Modeling using Constraint Graphs. 2021 IEEE/CVF Int. Conf. Comput. Vis., IEEE; 2021, p. 6670–80. https://doi.org/10.1109/ICCV48922.2021.00662.

[51] Nauata N, Hosseini S, Chang K-H, Chu H, Cheng C-Y, Furukawa Y. House-GAN++: Generative Adversarial Layout Refinement Network towards Intelligent Computational Agent for Professional Architects. 2021 IEEE/CVF Conf. Comput. Vis. Pattern Recognit., IEEE; 2021, p. 13627–36. https://doi.org/10.1109/CVPR46437.2021.01342.

[52] Klawitter J, Klesen F, Wolff A. Algorithms for Floor Planning with Proximity Requirements, 2022, p. 151–71. https://doi.org/10.1007/978-981-19-1280-1_10.

[53] Duarte JP. Customizing mass housing: a discursive grammar for Siza's Malagueira houses 2001.

[54] Müller P, Zeng G, Wonka P, Van Gool L. Image-based procedural modeling of facades. ACM Trans Graph 2007;26:85. https://doi.org/10.1145/1276377.1276484.

[55] Wu F, Yan D-M, Dong W, Zhang X, Wonka P. Inverse procedural modeling of facade layouts. ACM Trans Graph 2014;33:1–10. https://doi.org/10.1145/2601097.2601162.

[56] Rahbar M, Mahdavinejad M, Markazi AHD, Bemanian M. Architectural layout design through deep learning and agent-based modeling: A hybrid approach. J Build Eng 2022;47:103822. https://doi.org/10.1016/j.jobe.2021.103822.

[57] Wang L, Liu J, Zeng Y, Cheng G, Hu H, Hu J, et al. Automated building layout generation using deep learning and graph algorithms. Autom Constr 2023;154:105036. https://doi.org/10.1016/j.autcon.2023.105036.

[58] Weber RE, Mueller C, Reinhart C. Automated floorplan generation in architectural design: A review of methods and applications. Autom Constr 2022;140:104385. https://doi.org/10.1016/j.autcon.2022.104385.

[59] Watson KH, Giffin JW. The Vertex Splitting Algorithm for facilities layout. Int J Prod Res 1997;35:2477–92. https://doi.org/10.1080/002075497194615.

[60] Upasani N, Shekhawat K, Sachdeva G. Automated generation of dimensioned rectangular floorplans. Autom Constr 2020;113:103149. https://doi.org/10.1016/j.autcon.2020.103149.

[61] Shekhawat K, Jain RN, Bisht S, Kondaveeti A, Goswami D. Graph-based approach for enumerating floorplans based on users specifications. Artif Intell Eng Des Anal Manuf 2021;35:438–59. https://doi.org/10.1017/S0890060421000275.

[62] Wong DF, Liu CL. A New Algorithm for Floorplan Design. 23rd ACM/IEEE Des. Autom. Conf., IEEE; 1986, p. 101–7. https://doi.org/10.1109/DAC.1986.1586075.

[63] Seebohm T, Chan D. The design space of schematic Palladian plans for two villa topologies. Reinventing discourse-how Digit. tools Help Bridg. Transform Res. Educ. Pract. Archit. Proc. twenty first Annu. Conf. Assoc. Comput. Des. Archit., 2001, p. 156–65.